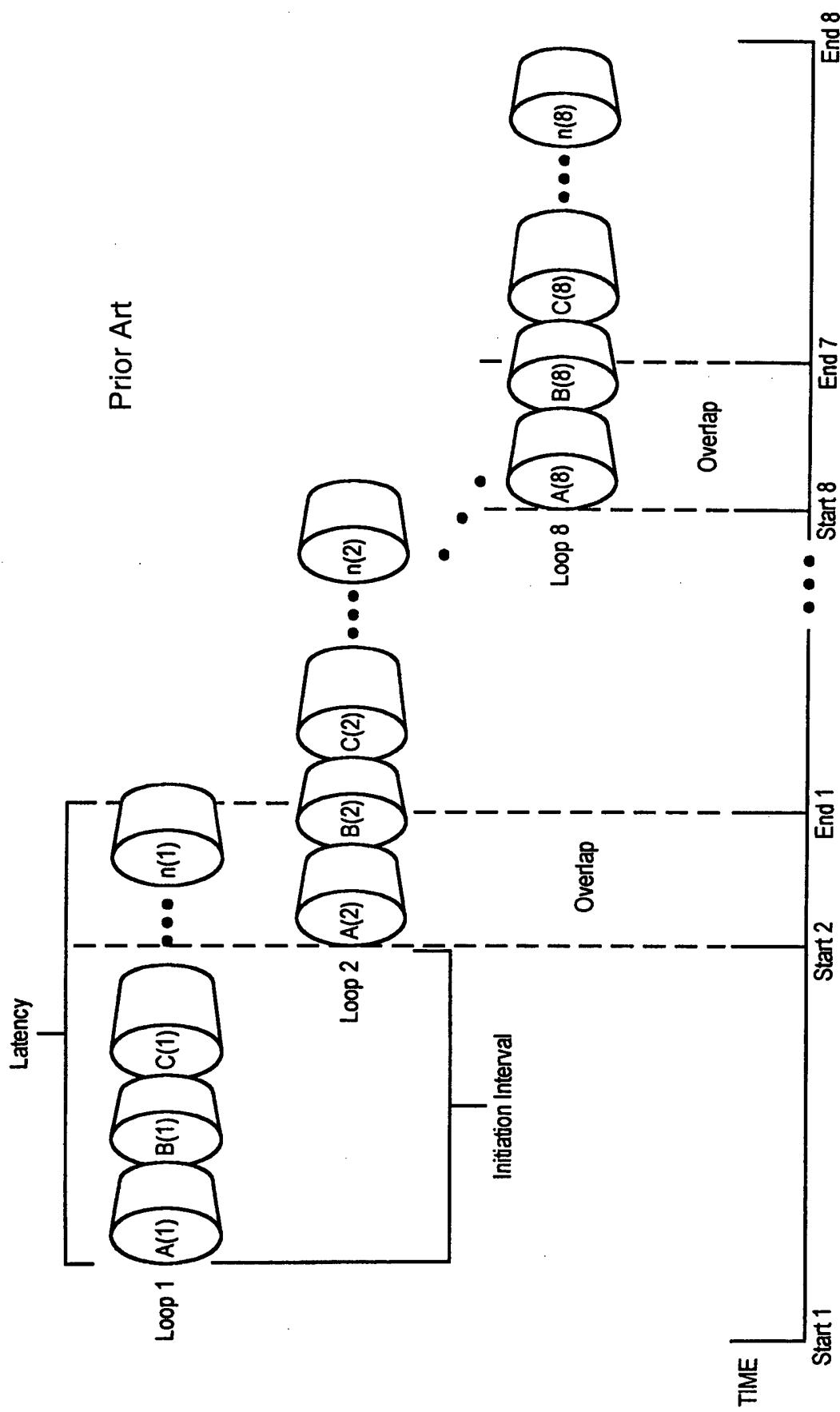


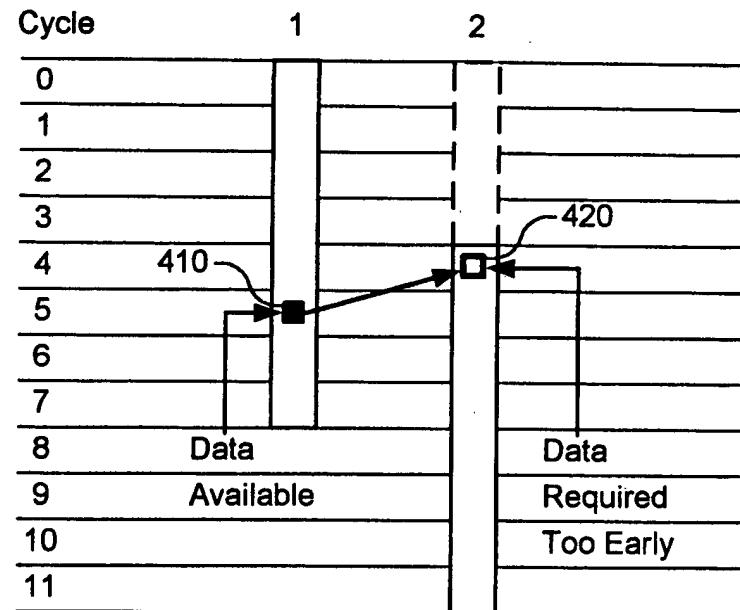
**Figure 1** Amended





Prior Art

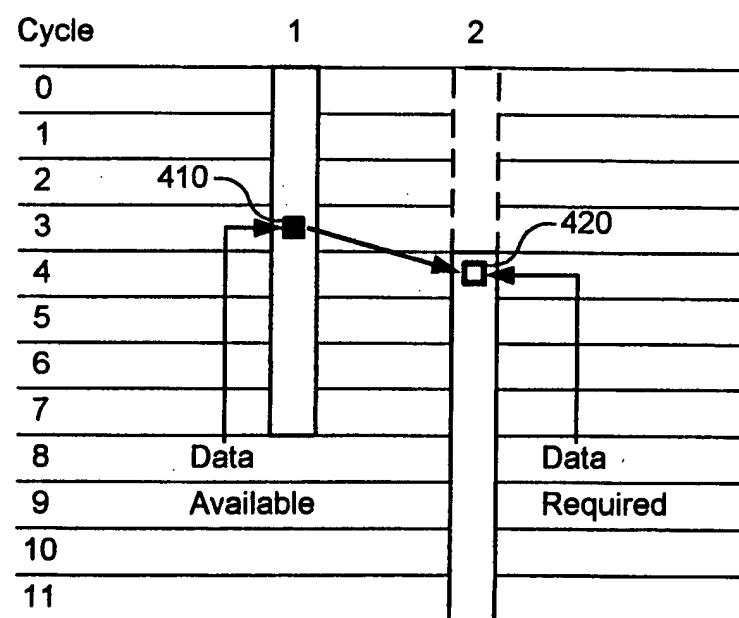
Loop Iterations



**Figure 3 (a)** Amended

Prior Art

Loop Iterations

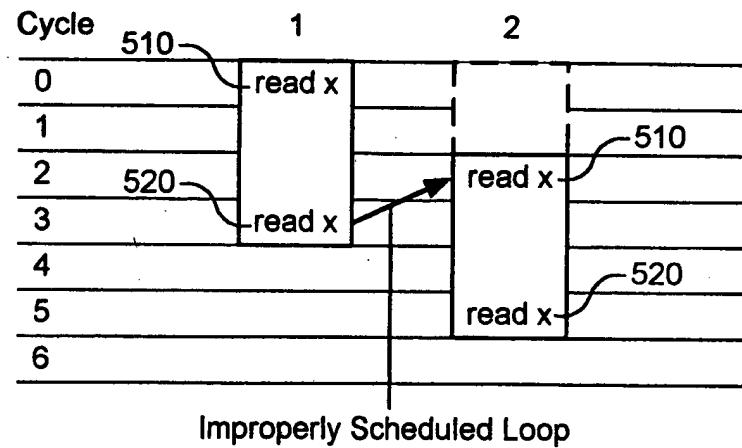


**Figure 3 (b)** Amended



Prior Art

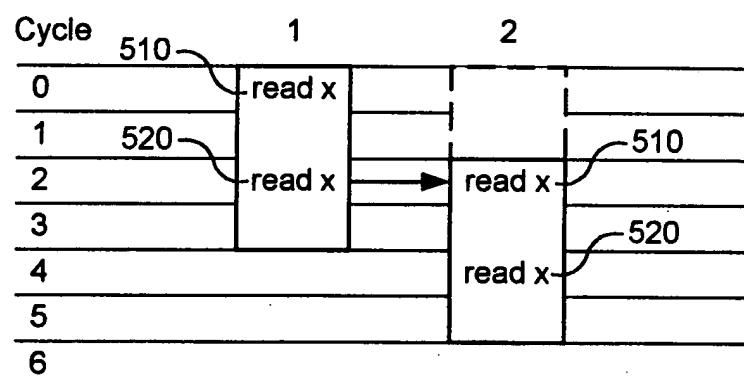
Loop Iterations



**Figure 4 (a)** Amended

Prior Art

Loop Iterations



**Figure 4 (b)** Amended

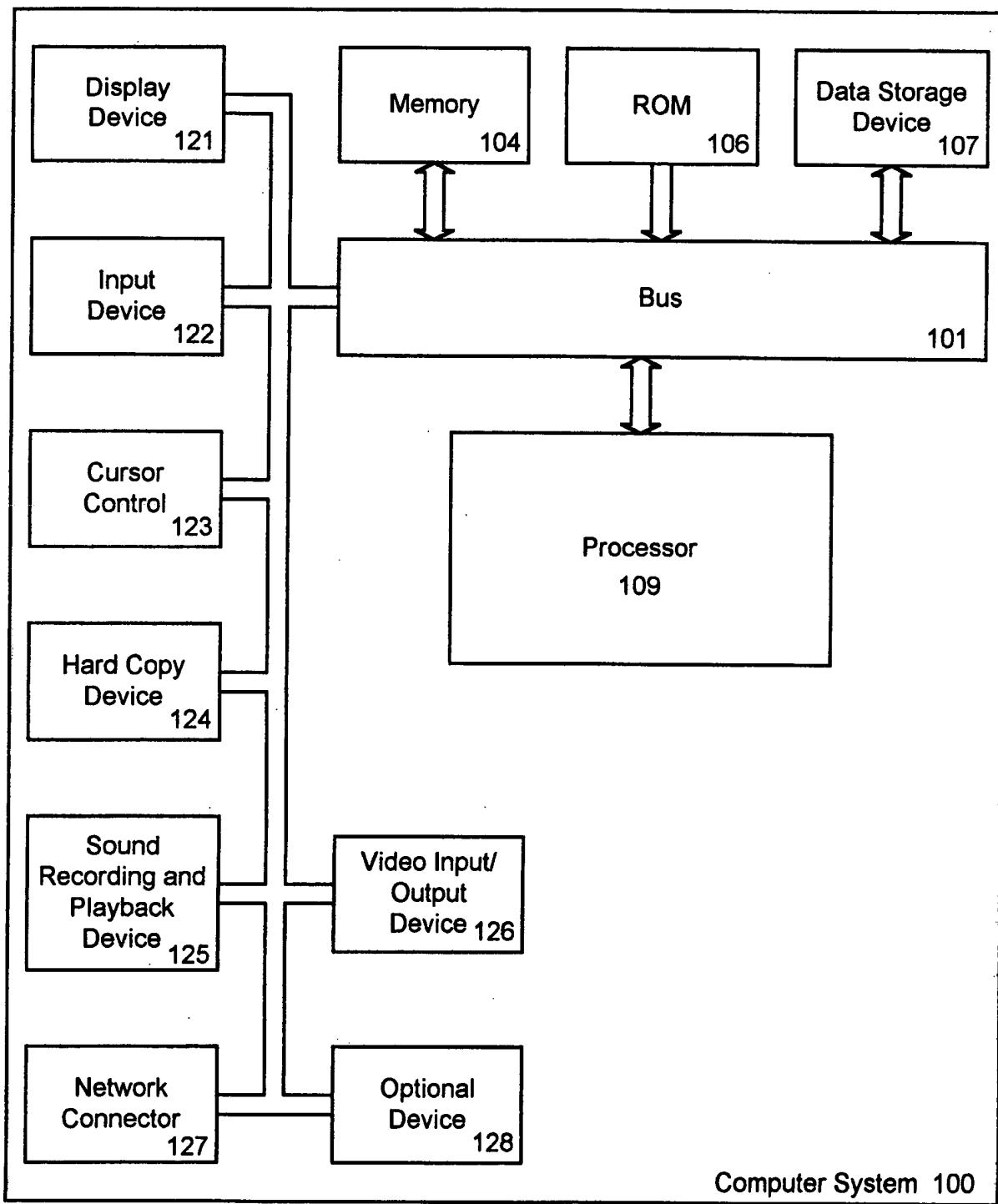
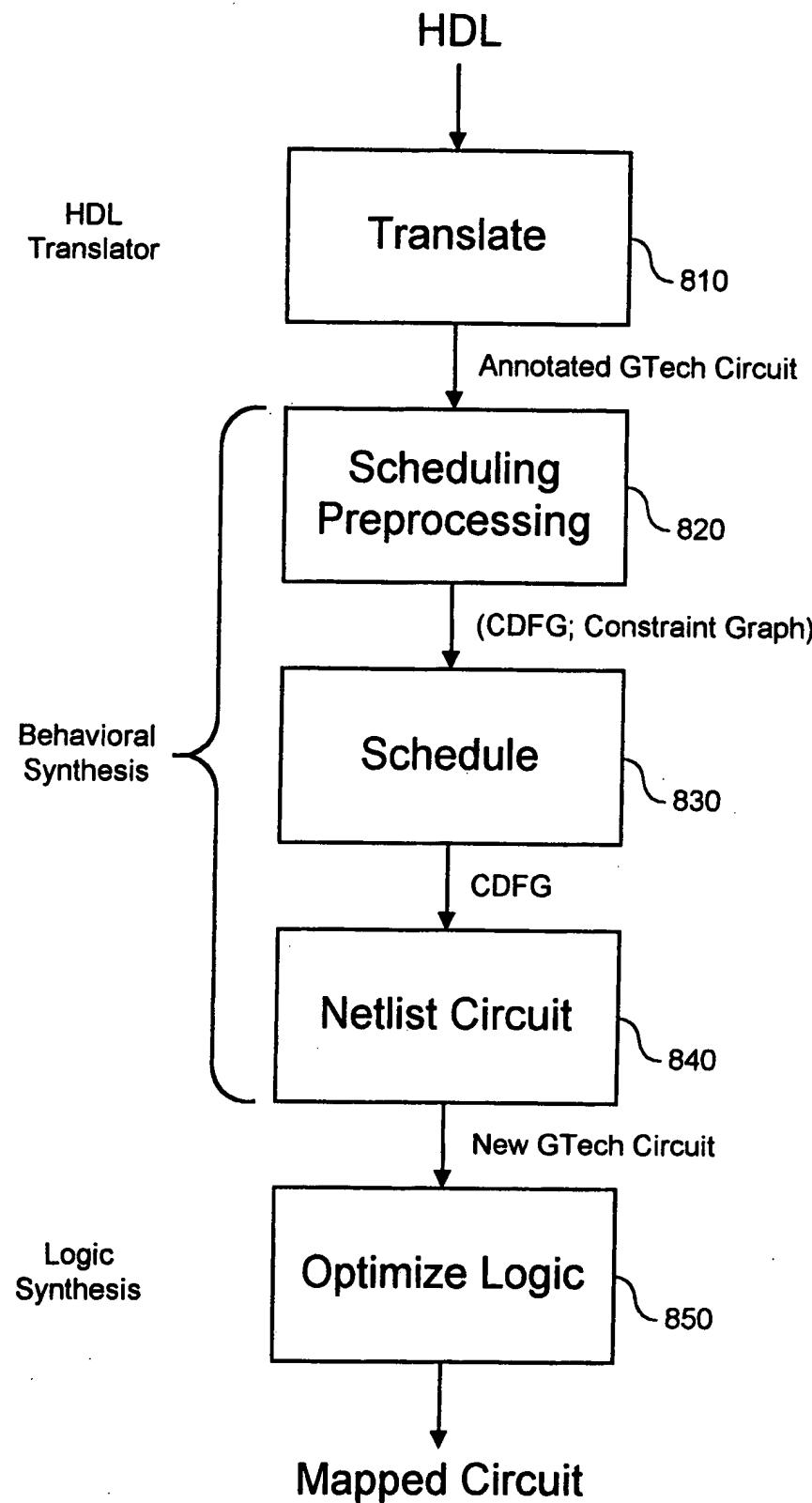
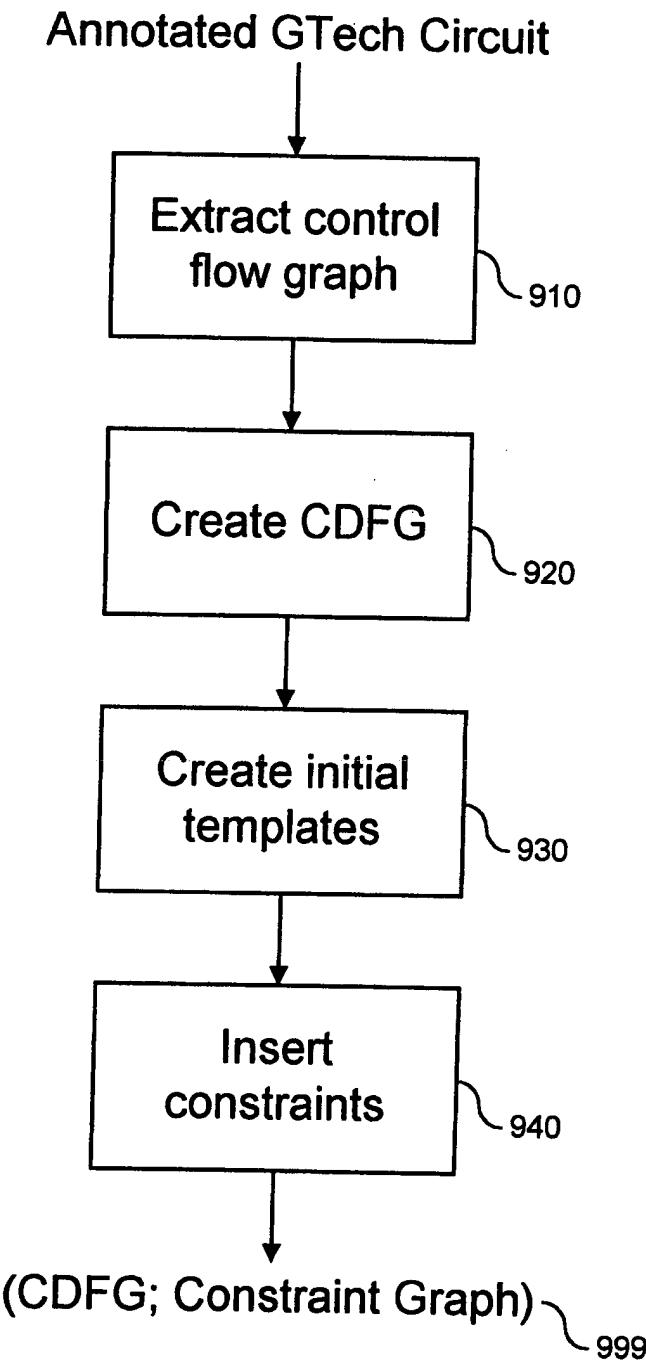


Figure 5



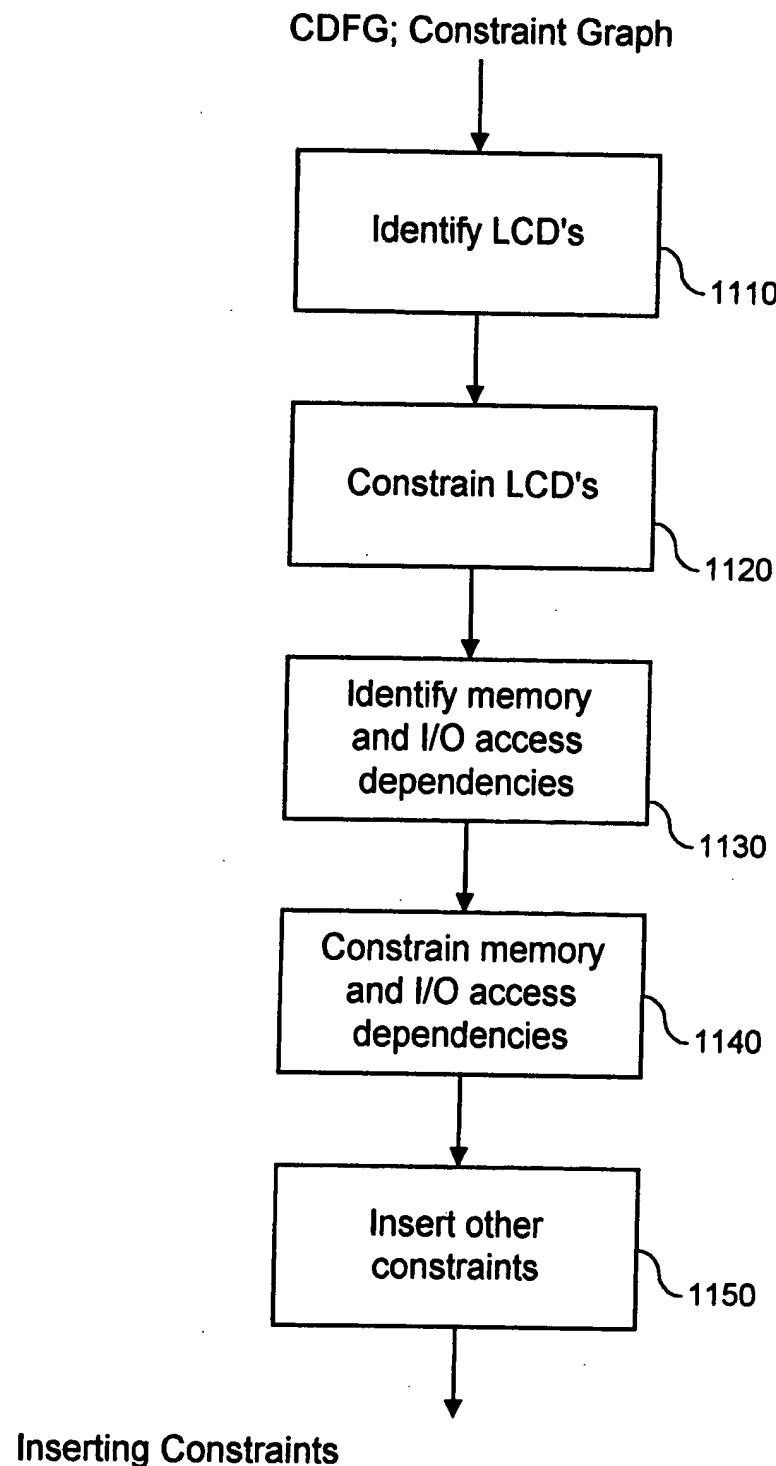
**Figure 6**

Synthesis with Scheduling

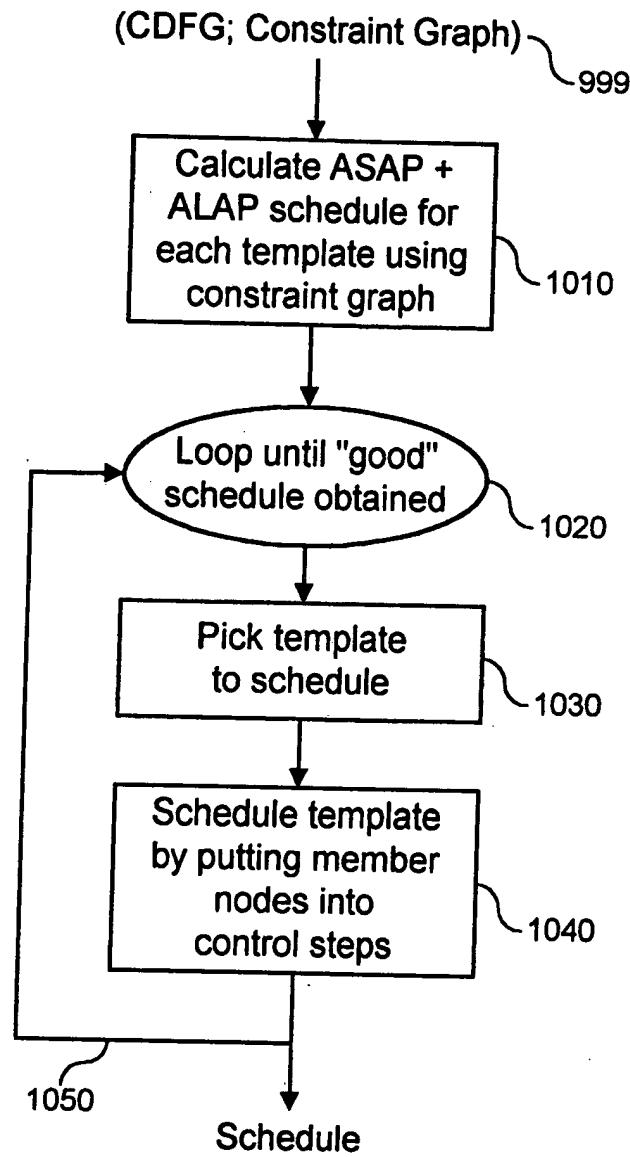


**Figure 7**

Scheduling  
Preprocessing

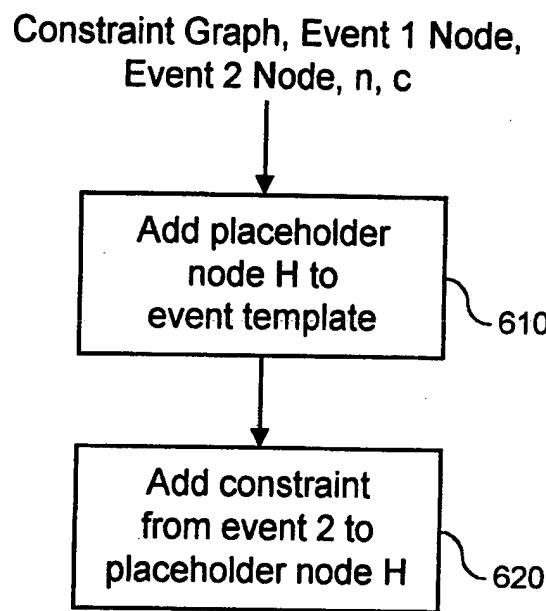


**Figure 8**



Scheduling Using Templates

**Figure 9**



**Figure 10**



```
module loopex8 ( c, x, y, z, clock);
  input [1:0] x, y, z;
  input clock ;
  output [2:0] c;
  reg [2:0] c;
  reg [2:0] p;

  always begin
    forever begin : theloop
      c <= x - p ;
      @(posedge clock) ;
      p = y + z ;
      @(posedge clock) ;
    end
  end
endmodule
```

**Figure 11**

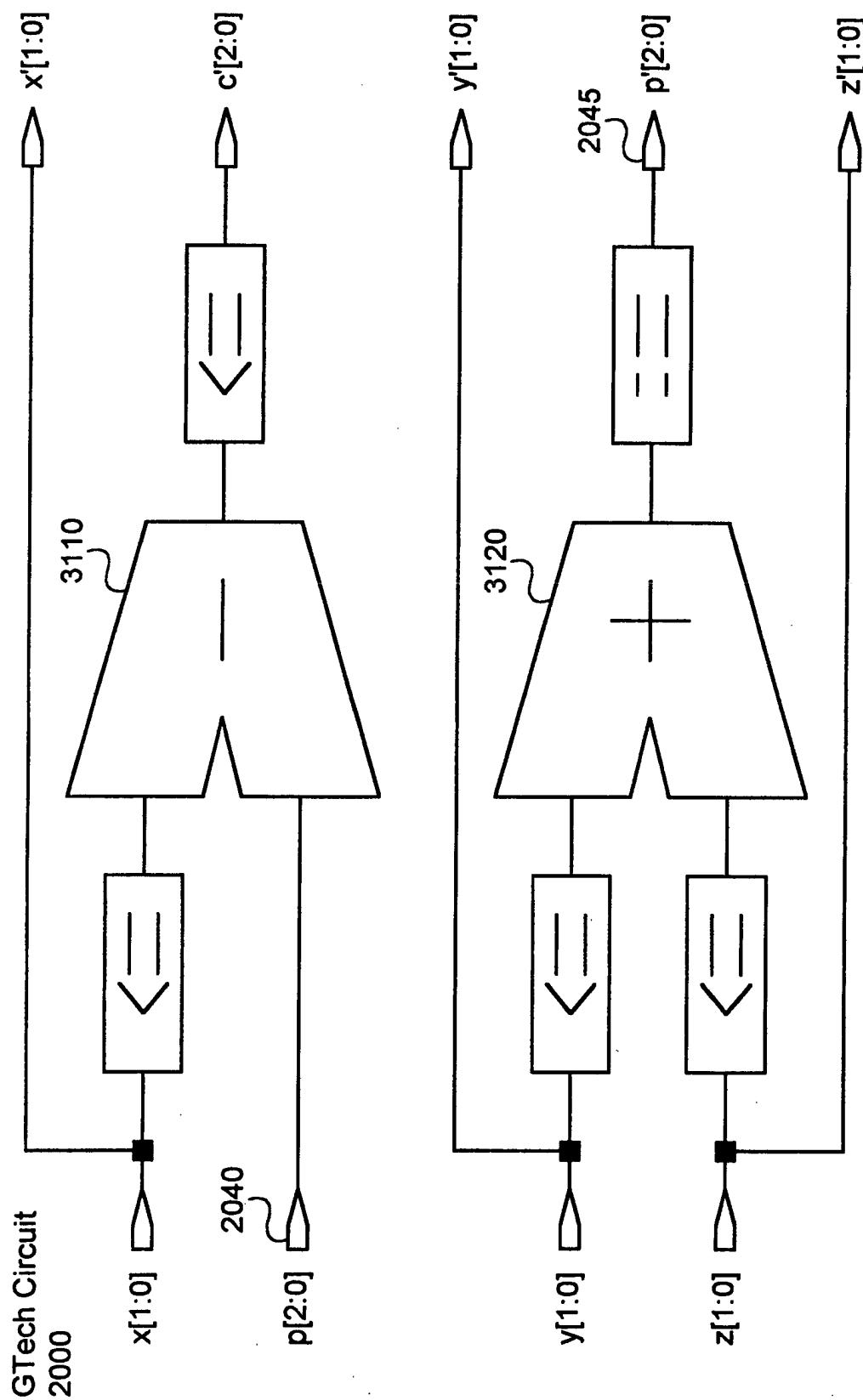
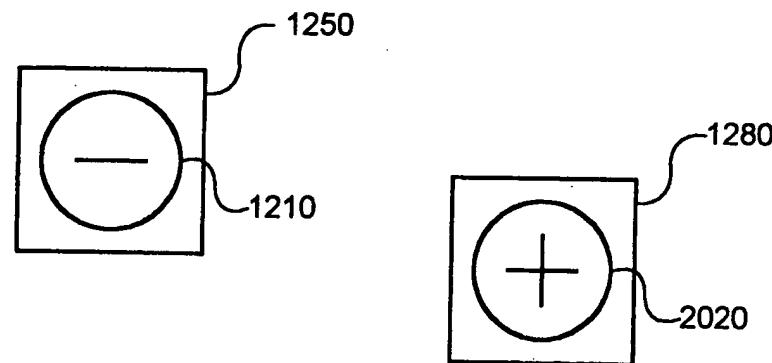
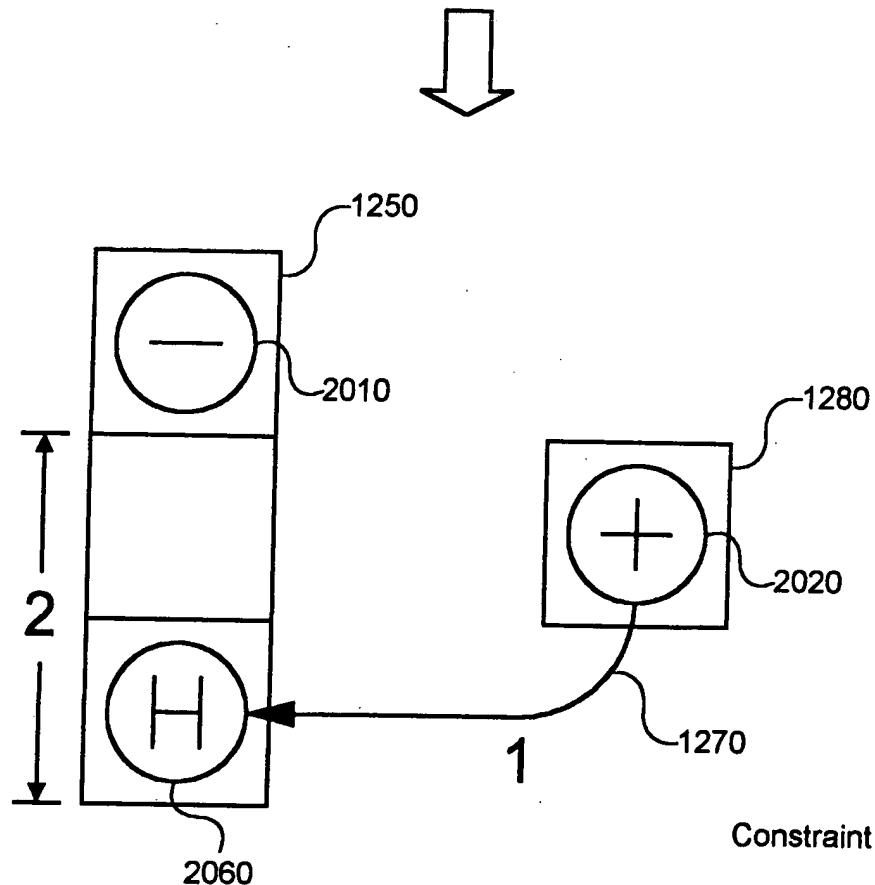


Figure 12



$$n = 2$$

**Figure 13a**



**Figure 13b**

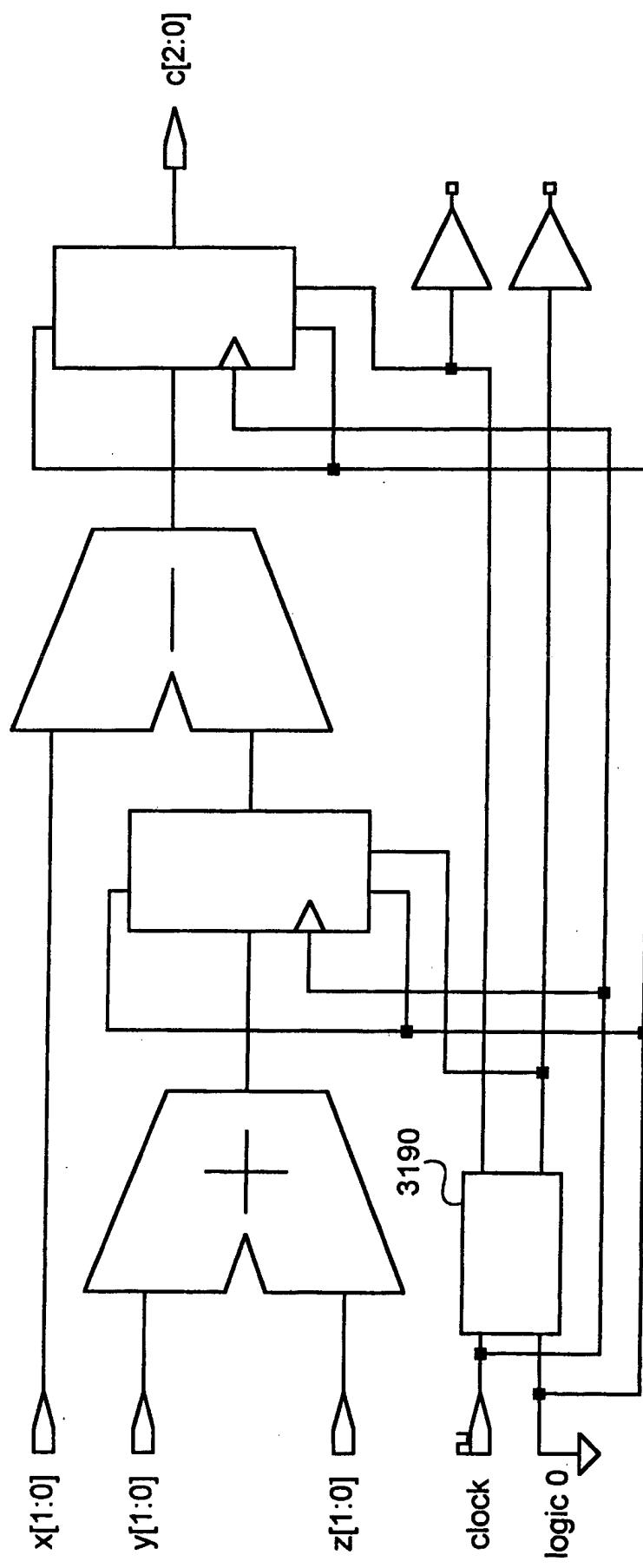


Figure 14



```
module write4 ( w, x, clock);  
  
    input [15:0] x ;  
    input clock ;  
    output [31:0] w;  
    reg [32:0] w;  
    reg [15:0] x1;  
    reg [15:0] x2;  
  
    always begin  
        forever begin : writeloop  
            x1 <= x ;  
            @(posedge clock) ;  
            x2 <= x ;  
            w <= x1 * x2 ;  
        end  
    end  
endmodule
```

**Figure 15**

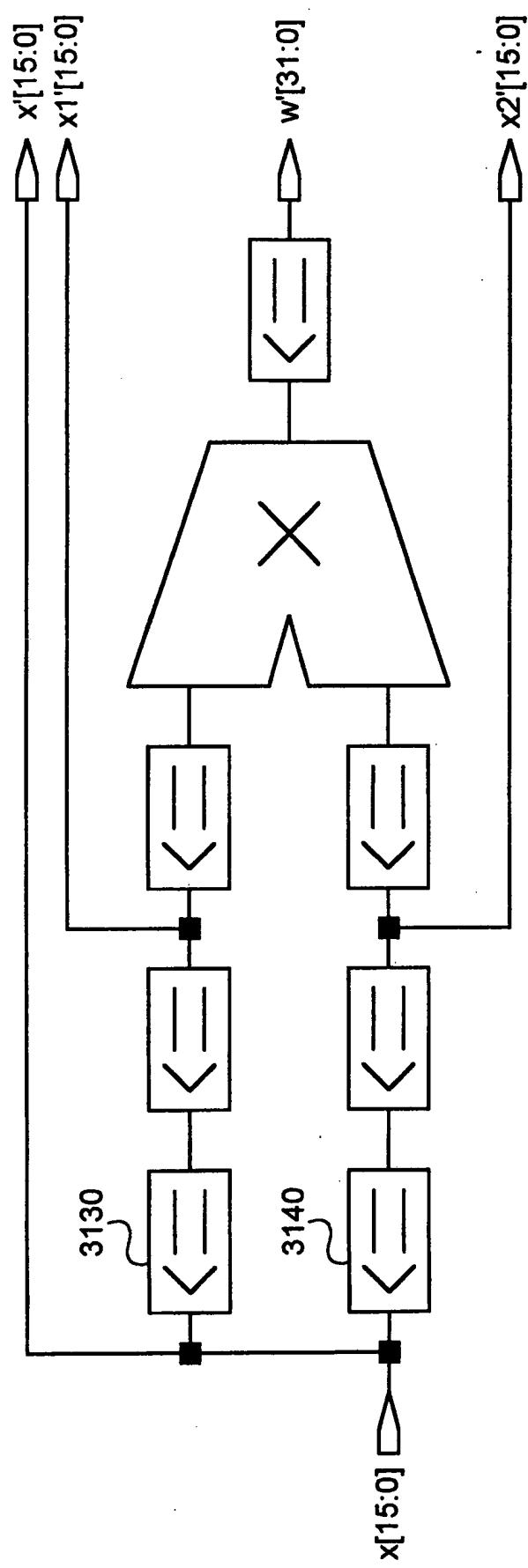
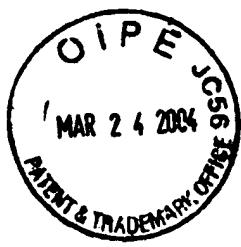
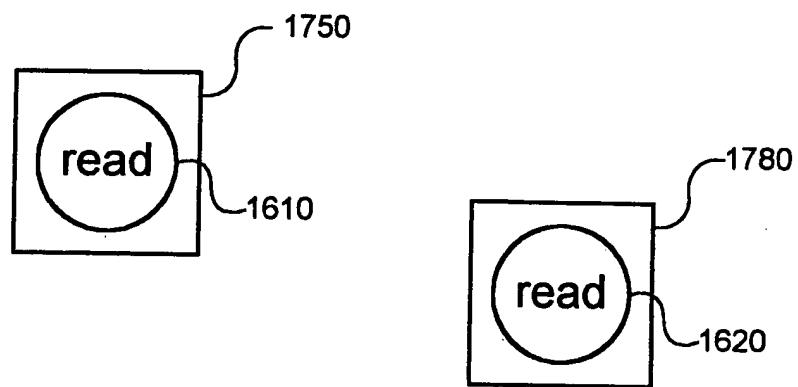
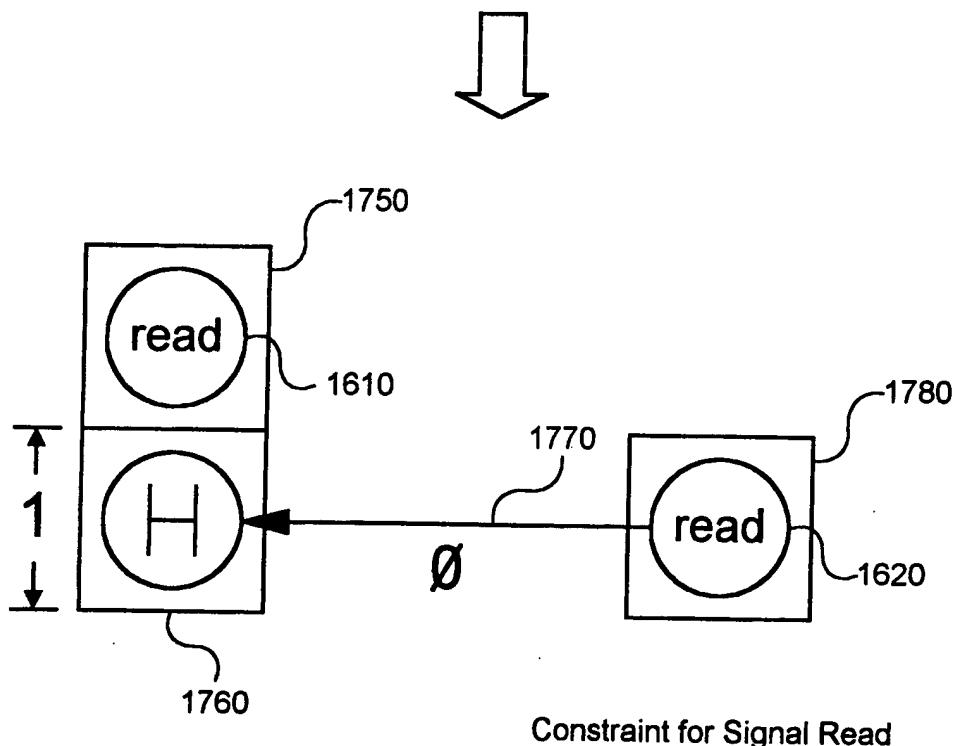


Figure 16



$n = 1$

**Figure 17a**



**Figure 17b**

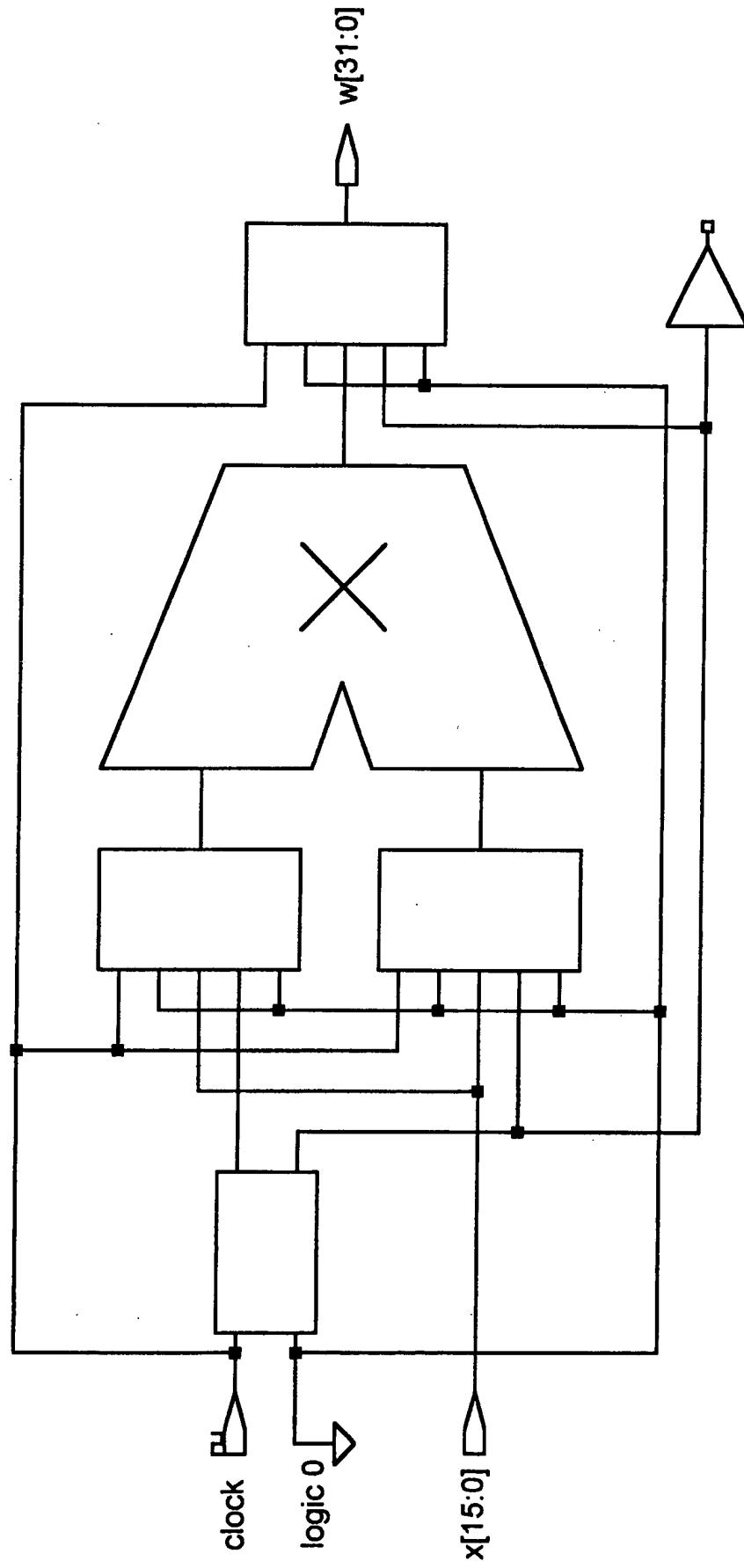


Figure 18



```
module after1 ( c, x, y, z, clock);

    input [1:0] x, y, z;
    input clock ;
    output [2:0] c;
    reg [2:0] c;
    reg [2:0] p;

    always begin

        @(posedge clock) ;

        forever begin
            c <= #24 x - p ;
            @(posedge clock) ;

            p = y + z ;
            @(posedge clock) ;
        end
    end
endmodule
```

**Figure 19 (a)**



```

entity after1 is
  port(
    c : out integer range 0 to 7;
    x, y, z : in integer range 0 to 3;
    clock : in bit
  );
end after1;

architecture behavioral of after1 is begin
  process
    variable p : integer range 0 to 7;
  begin
    wait until clock'event and clock = '1';

    loop
      c <= transport x - p after 24 ns;

      wait until clock'event and clock = '1';

      p := y + z;

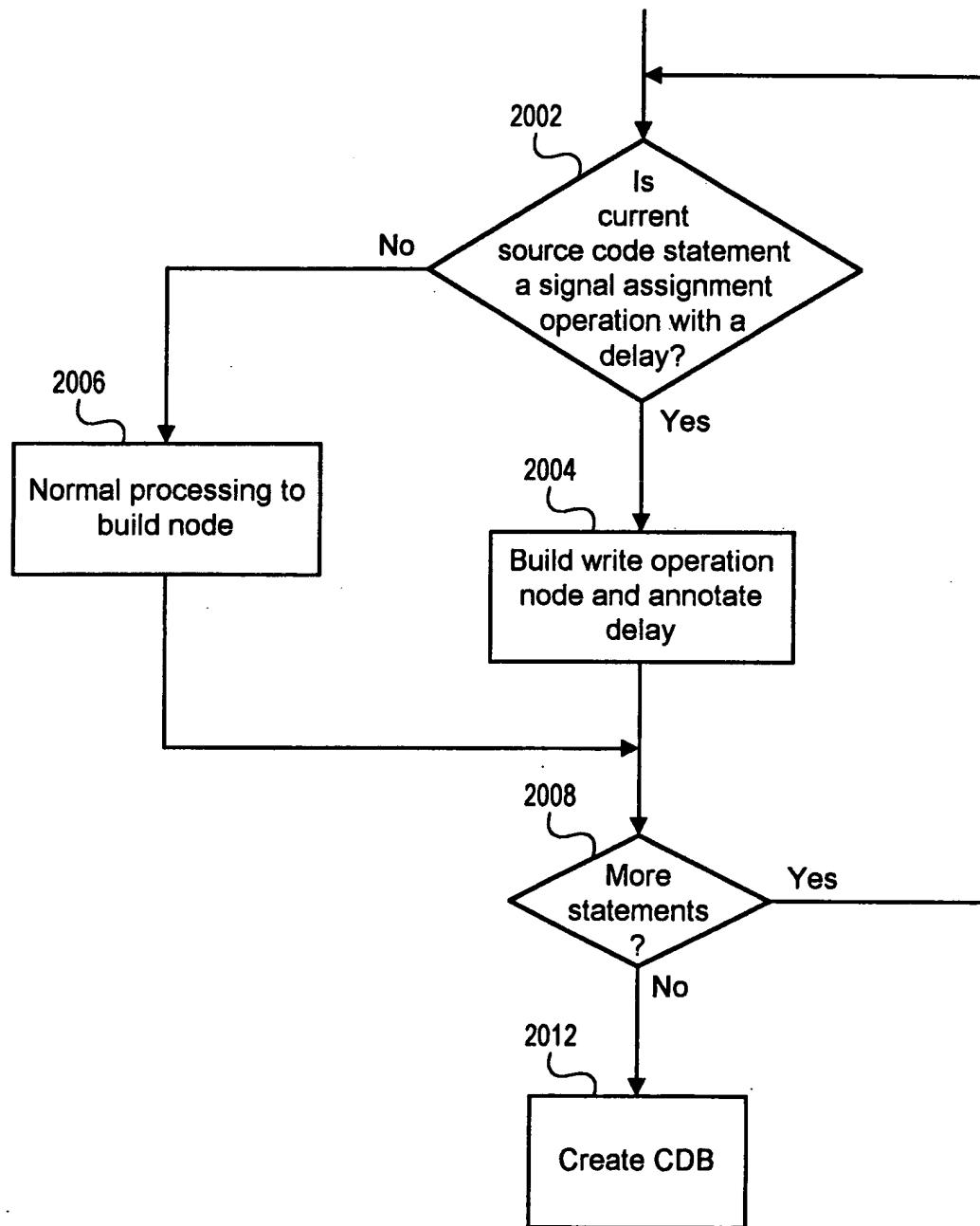
      wait until clock'event and clock = '1';

    end loop;

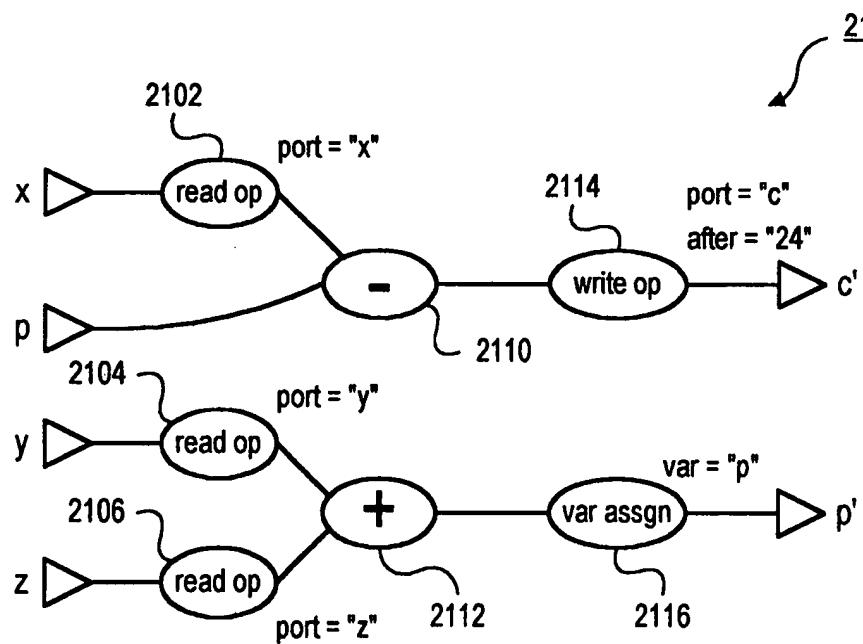
  end process;
end behavioral;

```

**Figure 19 (b)**

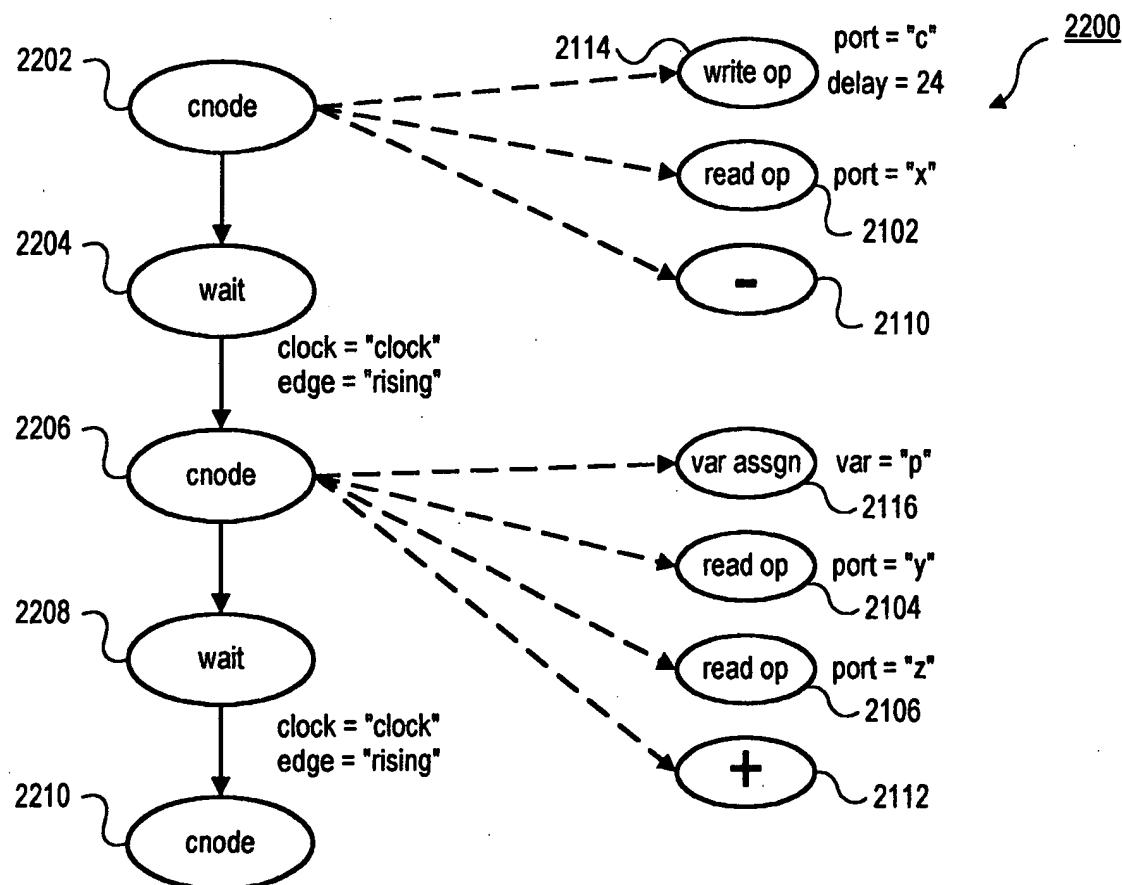


**Figure 20**



Data Flow Graph

Figure 21



CDB for forever loop  
(Control flow graph)

Figure 22

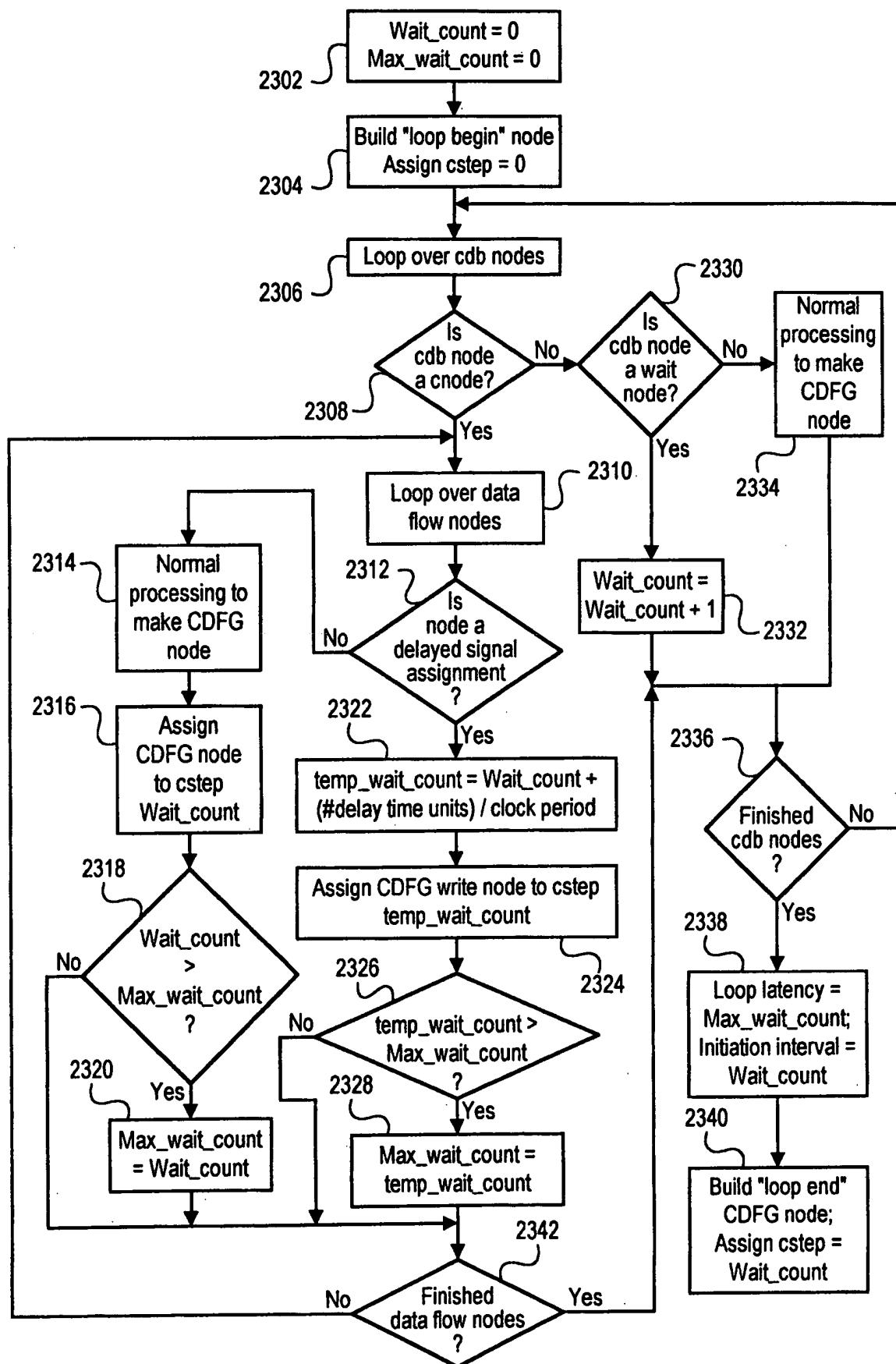


Figure 23

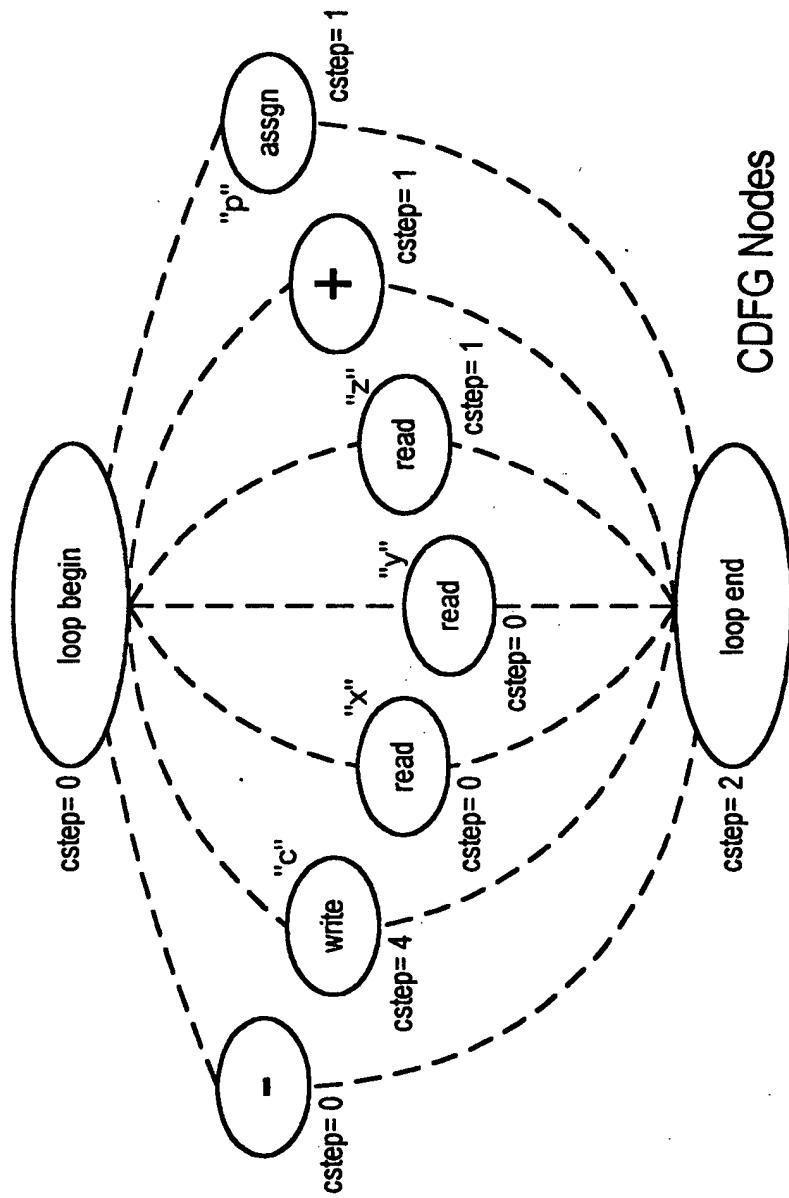


Figure 24

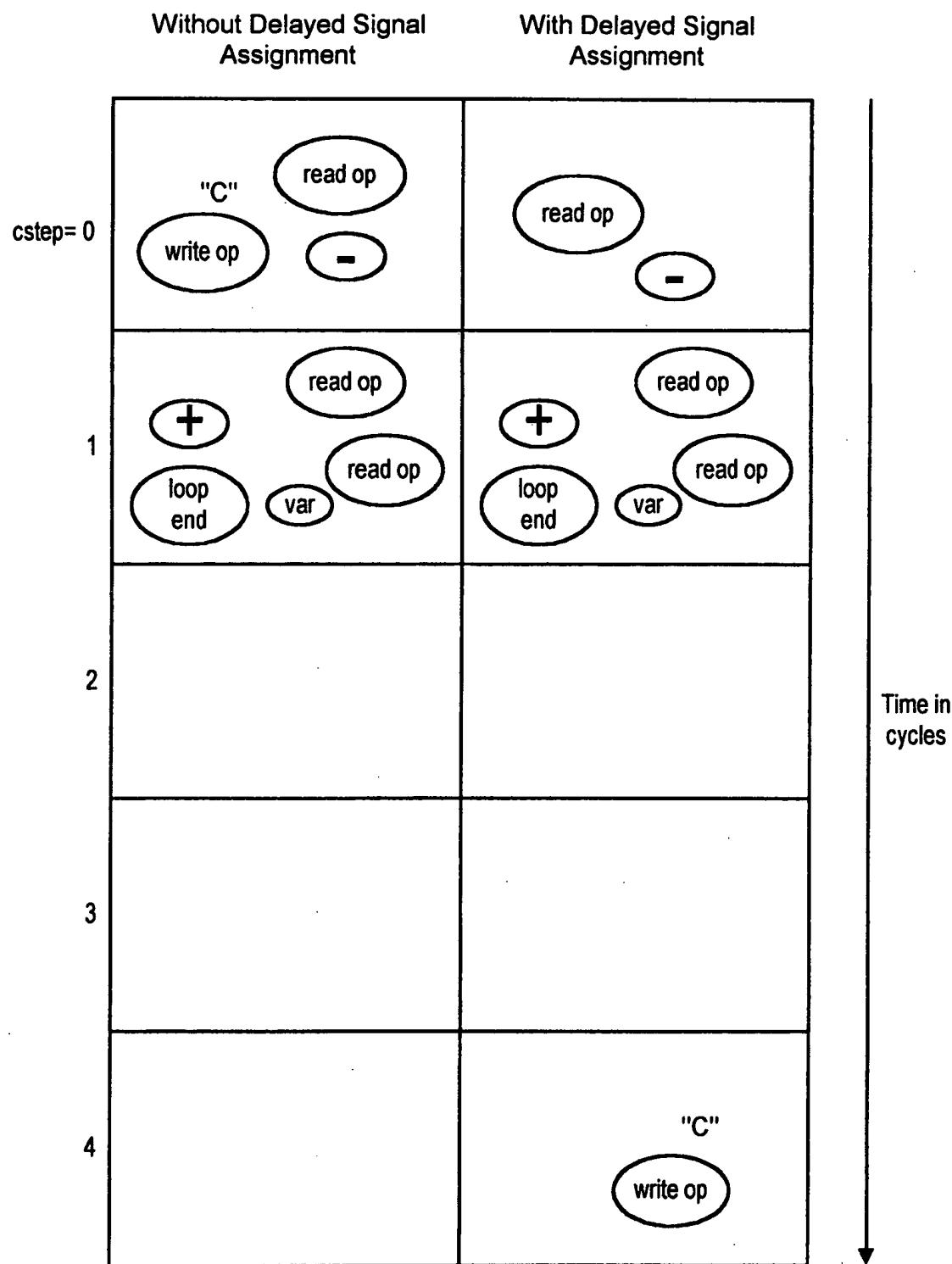


Figure 25

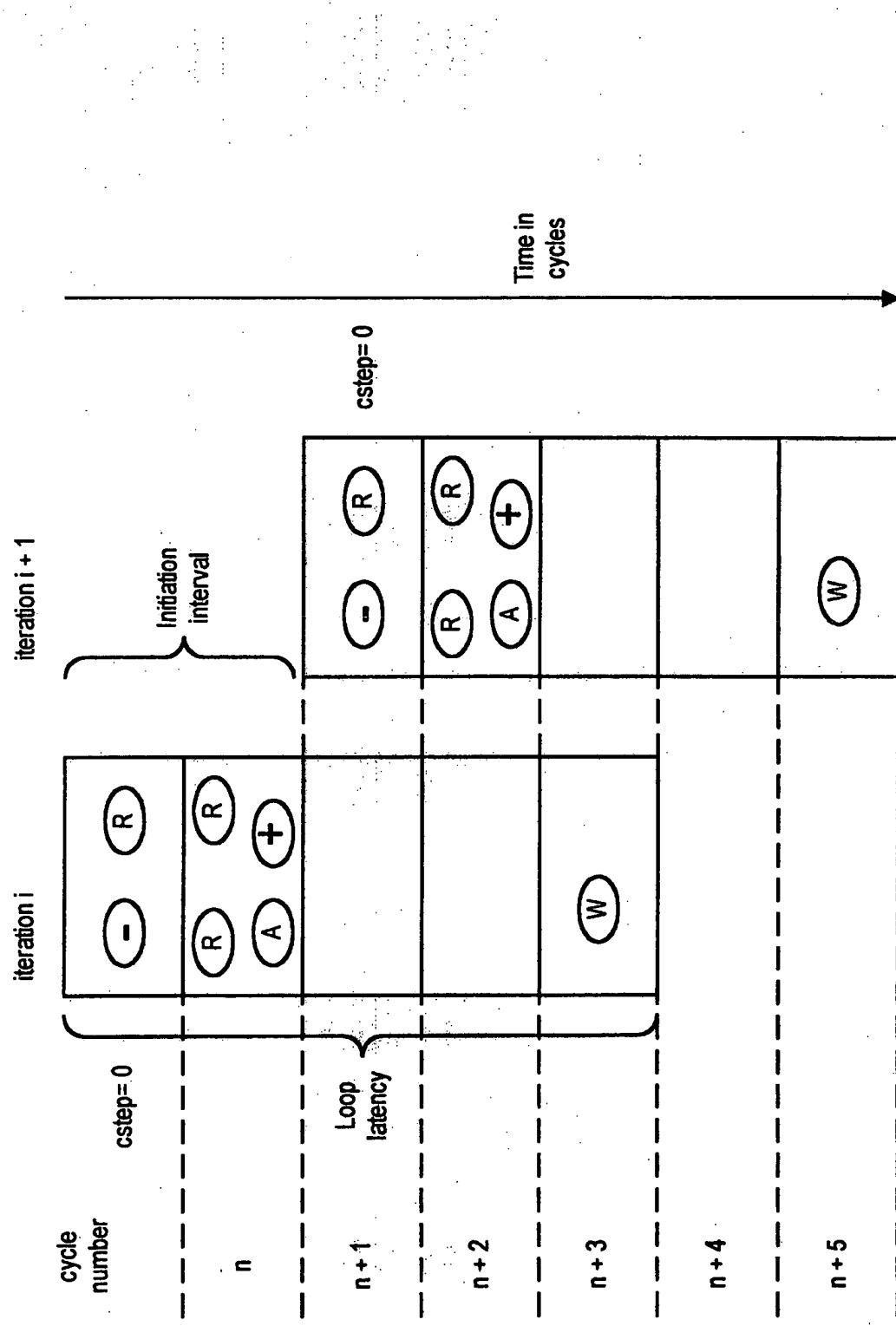
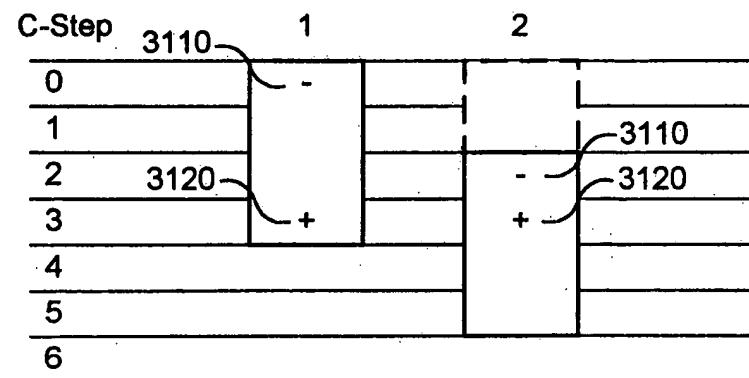


Figure 26

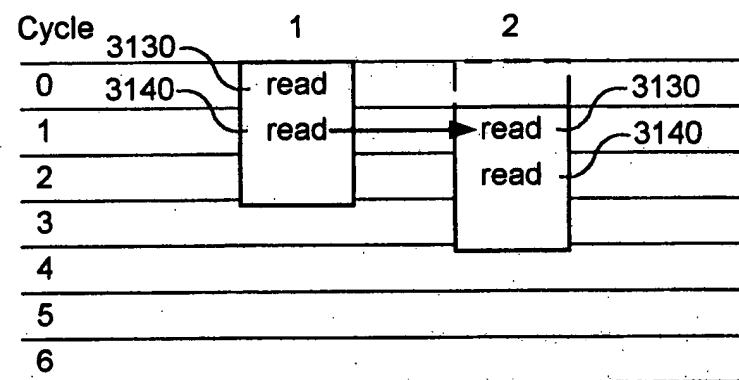


Loop Iterations

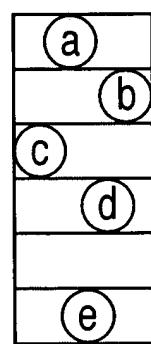
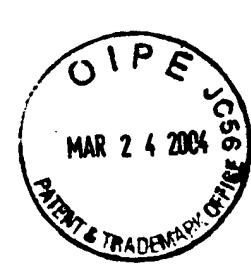


**Figure 27**

Loop Iterations

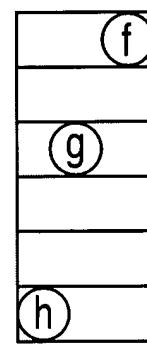


**Figure 28**



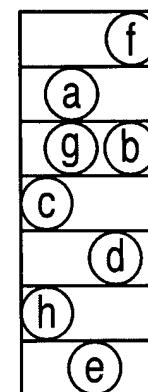
▀ cdfg behavioral template  
 ○ cdfg node

**FIG. 29a**



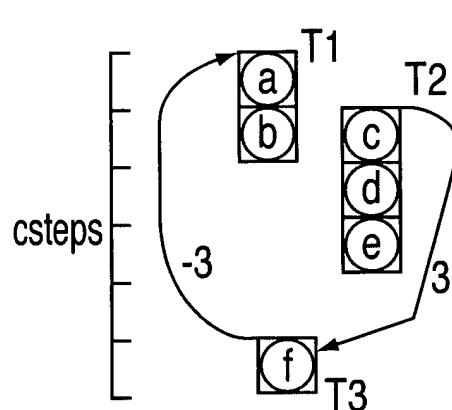
▀ cdfg behavioral template  
 ○ cdfg node

**FIG. 29b**



▀ cdfg behavioral template  
 ○ cdfg node

**FIG. 29c**

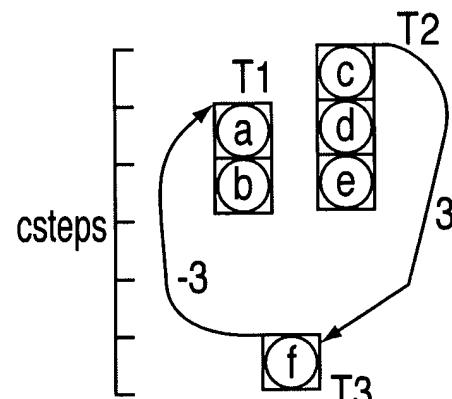


○ a uses  
 same  
 c resource

$$LP(T2, T1) = 0$$

**FIG. 30a**

New

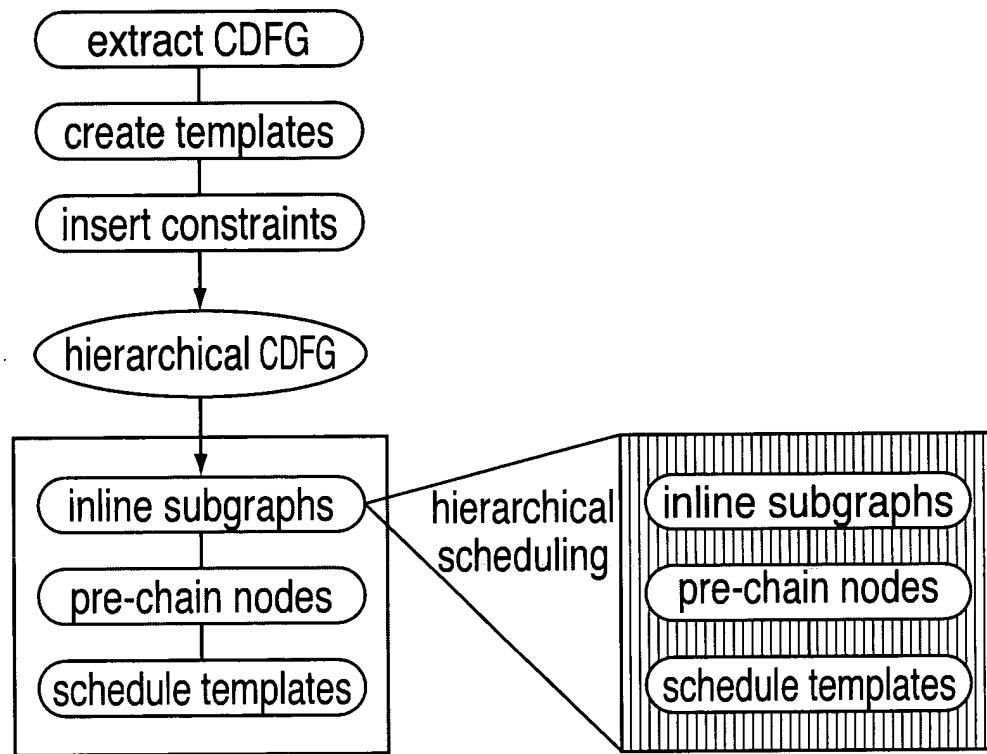


○ a uses  
 same  
 c resource

$$LP(T2, T1) = 0$$

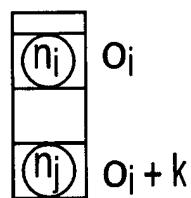
**FIG. 30b**

New



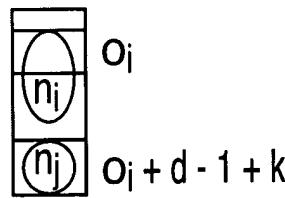
**FIG. 31**

New



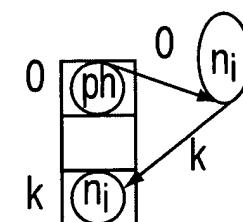
**FIG. 32a**

New



**FIG. 32b**

New



**FIG. 32c**

New

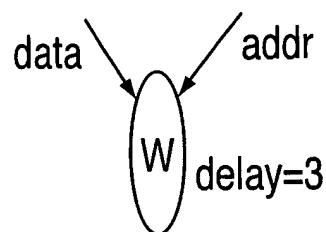


FIG. 33a

New

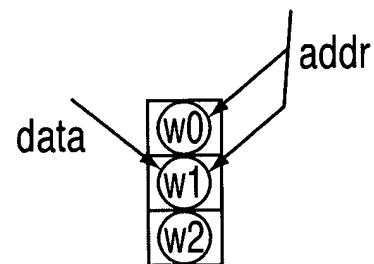


FIG. 33b

New

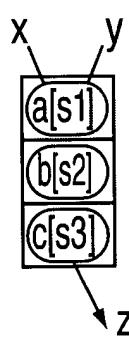


FIG. 34a

New

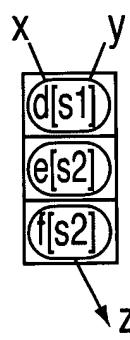


FIG. 34b

New

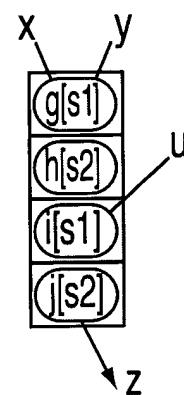


FIG. 34c

New

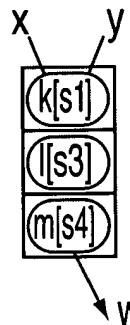


FIG. 34d

New

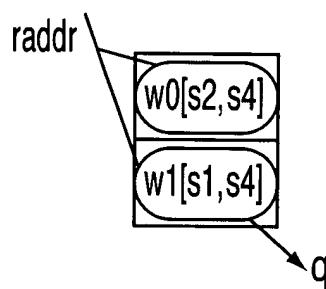


FIG. 35a New

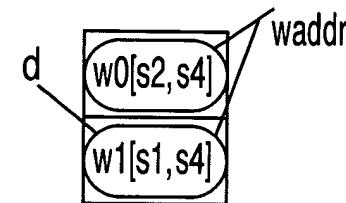
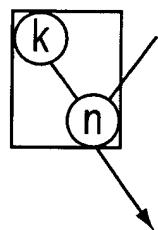
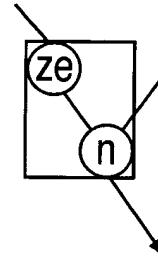


FIG. 35b New



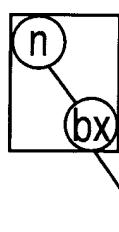
- (k) constant
- (ze) zero-extend
- (bx) bit-extract
- (u1) (u2) logic operations

FIG. 36a New



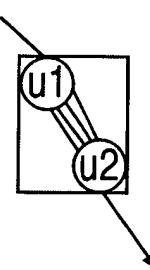
- (k) constant
- (ze) zero-extend
- (bx) bit-extract
- (u1) (u2) logic operations

FIG. 36b New



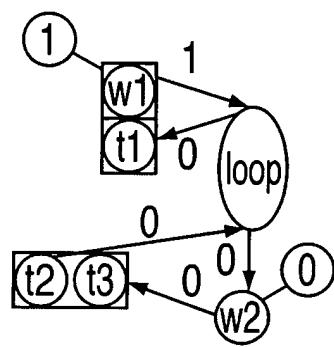
- (k) constant
- (ze) zero-extend
- (bx) bit-extract
- (u1) (u2) logic operations

FIG. 36c New



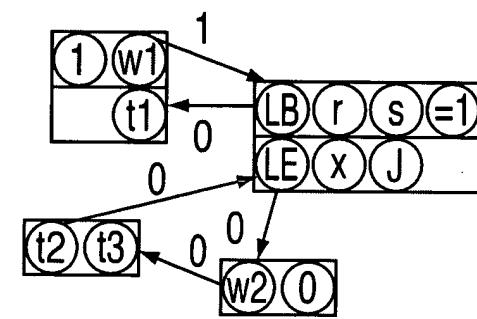
- (k) constant
- (ze) zero-extend
- (bx) bit-extract
- (u1) (u2) logic operations

FIG. 36d New



**FIG. 37a**

New



**FIG. 37b**

New

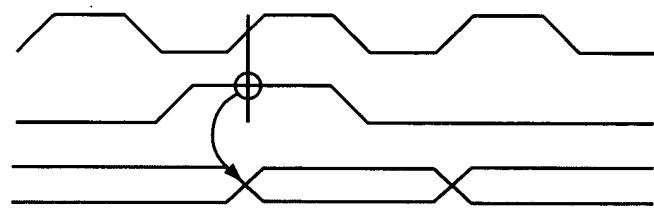


FIG. 38

New

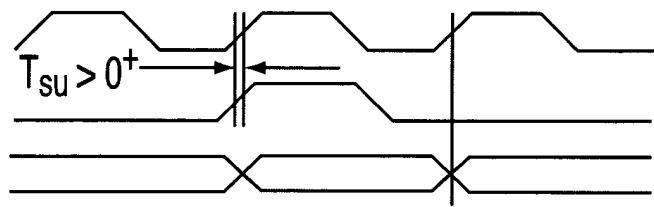


FIG. 39a

New

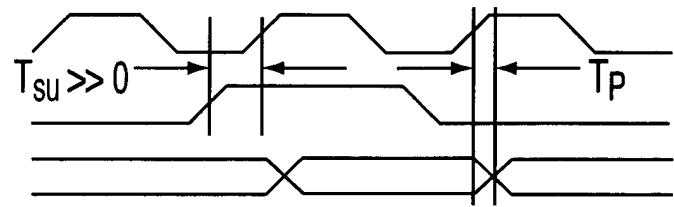


FIG. 39b

New



```

o1 <= v1;
while (c) begin: loop
  o2 <= v2;
  @(posedge clock);
end
o3 <= v3;
@(posedge clock);

```

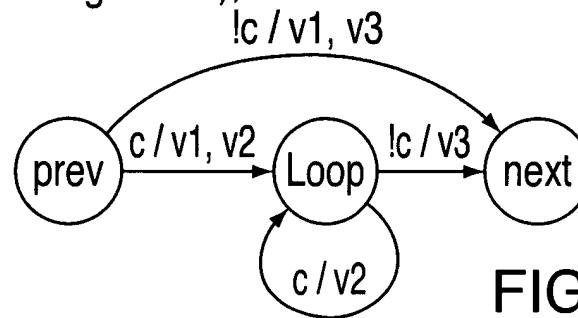


FIG. 40 New

```

o1 <= v1;
while (c) begin: loop
  @(posedge clock);
  o2 <= v2;
end
@(posedge clock);
o3 <= v3;

```

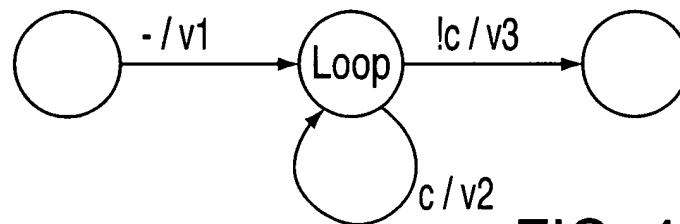


FIG. 41 New

```

@(posedge clock);
if (input_signal== 1'b1)begin
  x= input_read_1;
  y=input_read_2;
  tmp = x+ y;//2 cycle addition
  @(posedge clock);//strobe stab regs
  @(posedge clock);//1st cycle of add
  @(posedge clock);//2nd cycle of add
  out<=tmp;
end
@(posedge clock);

```

FIG. 42 New

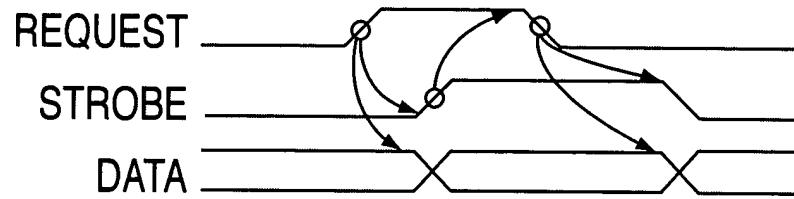


FIG. 43 New

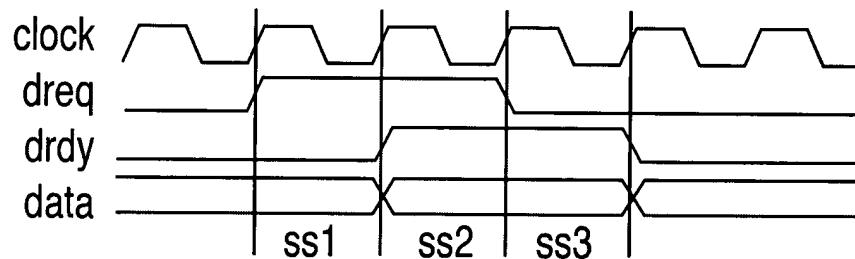


FIG. 44a New

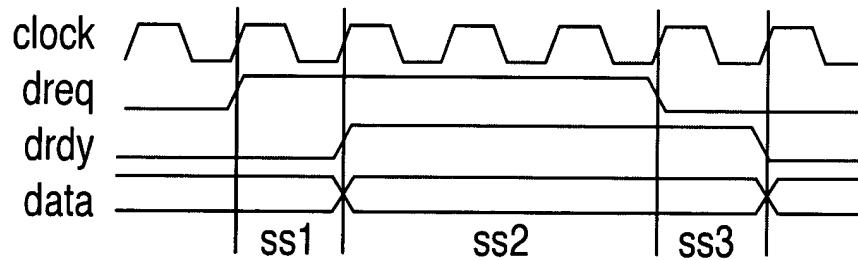


FIG. 44b New

```
a1=in_port1;  
a2=in_port2;  
@(posedge clock);  
out_port_1<=long_function_1(a1,a2);  
@(posedge clock);  
b1=in_port3;  
b2=in_port4;  
@(posedge clock);  
out_port_2<=long_function_2(b1,b2);
```

FIG. 45 New